

NAME

wait – synchronization with moving, counting and other activity

SYNOPSIS

```
wait(mode)
```

DESCRIPTION

The function `wait()` is used to synchronize the flow of commands in `spec` with moving, counting and other activity. Since the built-in commands and functions `move_all`, `move_cnt`, `tcount()` and `mcount()` return immediately after starting moving or counting, macros need to include some form of `wait()` if the next command requires the previous move or count to complete.

The argument `mode` specifies what is to be waited for:

- 0 – moving, counting and other acquisition
- 1 – (bit 0) moving
- 2 – (bit 1) counting (by the master timer)
- 4 – (bit 2) other acquisition (MCAs, CCDs, etc.)
- 8 – (bit 3) remote connections and remote asynchronous events

- 32 – (bit 5) return zero or nonzero to indicate if busy

If `mode` is a negative number, `wait()` will behave as for `mode = 0`, but a message will be printed showing what is being waited on.

For acquisition devices with "auto_run" mode enabled (such devices are started automatically during counting), waiting for counting will also include waiting for those devices (as of `spec` release 5.09.01-1).

When `spec` is running as client to a `spec` server, bit 3 checks if `remote_async()` replies have all arrived. In addition, bit 3 also checks if all configured `spec` servers have connected and if all `spec` server and EPICS remote motors have connected.

Waiting for `spec` server and remote motor connections is mainly an issue on start up or after reconfig. One might use `wait(8)` or `wait(0x28)` in the built-in special macro `config_mac` if it is important to delay until all connections are up. Note, until remote `spec` server and EPICS motors are fully connected and usable, the positions reported for those motors will be the last saved positions from `spec`'s `settings` file.

Also, note that `wait(0)` does not check for the remote events flagged by bit 3. To wait for remote events requires explicitly setting bit 3 in `mode`. Also, a `^C` interrupts a `wait(8)` but doesn't change the conditions that caused `wait(8)` to block. That is, the next `wait(8)` will still block if there are still pending connections.

If bit 5 (0x20) in `mode` is set, `wait()` returns true (1) if the activities flagged by bits 0, 1, 2 or 3 are still going on, otherwise `wait()` returns false (0).

EXAMPLES

```
def waitall    'wait(0)'
def waitmove  'wait(1)'
def waitcount 'wait(2)'
def w        'wait(0); beep'
```

SEE ALSO

w server epics